Big Data Examination

Roll No. - DS5B-2121

Question 1

Considering left as dependent variable in HR dataset, split the dataset according to your last digit of roll no. (Example: if your roll no is ending with 0, the ratio will be 70, 30; if your roll no is ending with 1, the ratio will be 71, 29; if your roll no is ending with 2, the ratio will be 72, 28; if your roll no is ending with 3, the ratio will be 73, 27 etc.). Determine the accuracy of the model.

Importing Pyspark Library

It is an interface for Apache Spark in Python that allows us to write Spark applications using Python APIs, but also provides the PySpark

In [ ]:
```
!pip install pyspark
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/publ
ic/simple/
Collecting pyspark
  Downloading pyspark-3.2.1.tar.gz (281.4 MB)
     |████████████████████████████████| 281.4 MB 29 kB/s
Collecting py4j==0.10.9.3
  Downloading py4j-0.10.9.3-py2.py3-none-any.whl (198 kB)
     |████████████████████████████████| 198 kB 42.5 MB/s
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.2.1-py2.py3-none-any.whl size=281853642
sha256=8b46831582fe33020b51646e24a86fc72a74a84b94240776aece0e2d97207751
  Stored in directory: /root/.cache/pip/wheels/9f/f5/07/7cd8017084dce4e93e84e92efd1e1d53
34db05f2e83bcef74f
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9.3 pyspark-3.2.1
```

Importing Library, Creating Session and Reading Data

In [ ]:
```
from pyspark.sql import SparkSession
session = SparkSession.builder.appName("HR_comma_Dataset").getOrCreate()
data = session.read.csv("HR comma.csv", header = True, inferSchema = True)
#we reassign value of __name__ (inbuilt variable) to "__main__" and main is used as entr
# else the value of name might be different
```

In [ ]:
```
data.show(10)
```

```
+------------------+---------------+--------------+--------------------+-----------------
--+-------------+----+---------------------+-----+------+
|satisfaction_level|last_evaluation|number_project|average_montly_hours|time_spend_compa
ny|Work_accident|left|promotion_last_5years|sales|salary|
+------------------+---------------+--------------+--------------------+-----------------
--+-------------+----+---------------------+-----+------+
|              0.38|           0.53|             2|                 157|
```

```
3|              0|    1|                     0|sales|    low|
|                 0.8|            0.86|                  5|                   262|
6|              0|    1|                     0|sales|medium|
|                0.11|            0.88|                  7|                   272|
4|              0|    1|                     0|sales|medium|
|                0.72|            0.87|                  5|                   223|
5|              0|    1|                     0|sales|    low|
|                0.37|            0.52|                  2|                   159|
3|              0|    1|                     0|sales|    low|
|                0.41|             0.5|                  2|                   153|
3|              0|    1|                     0|sales|    low|
|                 0.1|            0.77|                  6|                   247|
4|              0|    1|                     0|sales|    low|
|                0.92|            0.85|                  5|                   259|
5|              0|    1|                     0|sales|    low|
|                0.89|             1.0|                  5|                   224|
5|              0|    1|                     0|sales|    low|
|                0.42|            0.53|                  2|                   142|
3|              0|    1|                     0|sales|    low|
+------------------+---------------+--------------+--------------------+-----------------
--+-------------+----+---------------------+-----+------+
only showing top 10 rows
```

Check Null Values in columns

In [ ]:
```
from pyspark.sql.functions import isnan, when, count, col
data.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in data.columns])
```

```
+------------------+---------------+--------------+--------------------+-----------------
--+-------------+----+---------------------+-----+------+
|satisfaction_level|last_evaluation|number_project|average_montly_hours|time_spend_compa
ny|Work_accident|left|promotion_last_5years|sales|salary|
+------------------+---------------+--------------+--------------------+-----------------
--+-------------+----+---------------------+-----+------+
|                 0|              0|             0|                   0|
0|              0|    0|                    0|    0|     0|
+------------------+---------------+--------------+--------------------+-----------------
--+-------------+----+---------------------+-----+------+
```

There are no null values in the Dataset so we will move to Exploratory Data ANalysis

SHowing the Data

In [ ]:
```
data.columns
```

Out[ ]:
```
['satisfaction_level',
 'last_evaluation',
 'number_project',
 'average_montly_hours',
 'time_spend_company',
 'Work_accident',
 'left',
 'promotion_last_5years',
 'sales',
 'salary']
```

In [ ]:
```
data.printSchema()
```

```
root
 |-- satisfaction_level: double (nullable = true)
 |-- last_evaluation: double (nullable = true)
 |-- number_project: integer (nullable = true)
 |-- average_montly_hours: integer (nullable = true)
 |-- time_spend_company: integer (nullable = true)
 |-- Work_accident: integer (nullable = true)
 |-- left: integer (nullable = true)
 |-- promotion_last_5years: integer (nullable = true)
 |-- sales: string (nullable = true)
 |-- salary: string (nullable = true)
```

Importing Vector Assembler, String Indexer and One Hot Encoder

In [ ]:
```
from pyspark.ml.feature import VectorAssembler, StringIndexer, OneHotEncoder
# It is use for mapping a string columm to a index column that will be treated as a cate
str_idx = StringIndexer(inputCols = ['sales','salary'], outputCols = ["newsales", "newsa
```

Applying OneHotEncoder and converting into 0,1 matrices

In [ ]:
```
# It is an important technique for converting categorical attributes into a numeric vect
one_hot = OneHotEncoder(inputCols = ["newsales","newsalary"], outputCols = ["newsales_on
```

In [ ]:
```
# It is feature transformer that combine multiple columns into a single vector column.
# Pyspark ml models takes only one independent variable and one dependent varibale
#but, we have multiple independent variabales, so we use vector assembler to convert the
# of independent variables
vec_ass = VectorAssembler(inputCols = ['satisfaction_level','last_evaluation','number_pr
```

In [ ]:
```
from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression(featuresCol= "all_features", labelCol = "left")
```

Creating Pipeline

Its like deciding the order of steps to be executed

In [ ]:
```
from pyspark.ml import Pipeline
mypipeline = Pipeline(stages = [str_idx, one_hot, vec_ass, lr])
```

Splitting the Dataset

As my roll no is DS5B-2121 I will be using split as 0.71 and 0.29

In [ ]:
```
training, test = data.randomSplit([0.71, 0.29])
```

Building the Model

```
In [ ]: lr_model = mypipeline.fit(training)
```

Fitting the data to the model

to compute patterns using Train data and then these will be applied on test data

```
In [ ]: result = lr_model.transform(test)
```

SHowing the Result

```
In [ ]: result.show(4, truncate = False)
```

```
+------------------+----------------+---------------+---------------------+----------------
--+-------------+----+---------------------+------------+------+--------+---------+------
---------+----------------+-------------------------------------------------------+----
-------------------------------------+-----------------------------------------+----------
+
|satisfaction_level|last_evaluation|number_project|average_montly_hours|time_spend_compa
ny|Work_accident|left|promotion_last_5years|sales       |salary|newsales|newsalary|newsal
es_onehot|newsalary_onehot|all_features                                           |rawP
rediction                                |probability                              |prediction
|
+------------------+----------------+---------------+---------------------+----------------
--+-------------+----+---------------------+------------+------+--------+---------+------
---------+----------------+-------------------------------------------------------+----
-------------------------------------+-----------------------------------------+----------
+
|0.09              |0.62            |6              |294                  |4
  |0            |1   |0                    |accounting |low   |8.0     |0.0      |(9,
[8],[1.0])  |(2,[0],[1.0])   |(18,[0,1,2,3,4,15,16],[0.09,0.62,6.0,294.0,4.0,1.0,1.0])|
[-0.906113009228541,0.906113009228541]  |[0.28779589386338533,0.7122041061366147]|1.0
    |
|0.09              |0.62            |6              |294                  |4
  |0            |1   |0                    |accounting |low   |8.0     |0.0      |(9,
[8],[1.0])  |(2,[0],[1.0])   |(18,[0,1,2,3,4,15,16],[0.09,0.62,6.0,294.0,4.0,1.0,1.0])|
[-0.906113009228541,0.906113009228541]  |[0.28779589386338533,0.7122041061366147]|1.0
    |
|0.09              |0.77            |5              |275                  |4
  |0            |1   |0                    |product_mng|medium|4.0     |1.0      |(9,
[4],[1.0])  |(2,[1],[1.0])   |(18,[0,1,2,3,4,11,17],[0.09,0.77,5.0,275.0,4.0,1.0,1.0])|
[-0.6706336468375675,0.6706336468375675]|[0.3383549711925885,0.6616450288074115] |1.0
    |
|0.09              |0.77            |6              |244                  |4
  |0            |1   |0                    |product_mng|low   |4.0     |0.0      |(9,
[4],[1.0])  |(2,[0],[1.0])   |(18,[0,1,2,3,4,11,16],[0.09,0.77,6.0,244.0,4.0,1.0,1.0])|
[-0.7998272517122431,0.7998272517122431]|[0.31006247261884357,0.6899375273811564]|1.0
    |
+------------------+----------------+---------------+---------------------+----------------
--+-------------+----+---------------------+------------+------+--------+---------+------
---------+----------------+-------------------------------------------------------+----
-------------------------------------+-----------------------------------------+----------
+
only showing top 4 rows
```

Evaluating the Logistic Regression Model using MultiClassificationEvaluator

As the number of unique values the dependent variable could take are more than 2, we have to apply MultiClassificationEvaluator insted of BinaryClassificationEvaluator

```
In [ ]: evaluation = ["f1","accuracy","weightedPrecision","weightedRecall", "weightedTruePositiv
        for i in evaluation:
          from pyspark.ml.evaluation import MulticlassClassificationEvaluator
          eval = MulticlassClassificationEvaluator(predictionCol="prediction", labelCol= "left",
          print(i, ":", eval.evaluate(result))
```

```
f1 : 0.7779568868738669
accuracy : 0.801658604008293
weightedPrecision : 0.7820065880658045
weightedRecall : 0.8016586040082929
weightedTruePositiveRate : 0.8016586040082929
weightedFalsePositiveRate : 0.5166086426447655
weightedFMeasure : 0.7779568868738669
truePositiveRateByLabel : 0.9422208847427024
falsePositiveRateByLabel : 0.6571709233791748
precisionByLabel : 0.8239473684210527
recallByLabel : 0.9422208847427024
fMeasureByLabel : 0.8791239646216482
logLoss : 0.4152158353029982
hammingLoss : 0.19834139599170697
```

Building the Decision Tree Classifier Model

```
In [ ]: from pyspark.ml.classification import DecisionTreeClassifier
        dtc = DecisionTreeClassifier(featuresCol= "all_features", labelCol = "left")
```

Creating Pipeline

```
In [ ]: dtc_model = mypipeline.fit(training)
```

```
In [ ]: result2 = dtc_model.transform(test)
```

Tranforming Data to compute the dataset

```
In [ ]: result2.show(4, truncate = False)
```

```
+------------------+---------------+--------------+--------------------+----------------
--+-------------+----+--------------------+-----------+------+--------+---------+------
---------+----------------+--------------------------------------------------------+----
-----------------------------------+----------------------------------------+----------
+
|satisfaction_level|last_evaluation|number_project|average_montly_hours|time_spend_compa
ny|Work_accident|left|promotion_last_5years|sales      |salary|newsales|newsalary|newsal
es_onehot|newsalary_onehot|all_features                                            |rawP
rediction                          |probability                             |prediction
|
+------------------+---------------+--------------+--------------------+----------------
--+-------------+----+--------------------+-----------+------+--------+---------+------
---------+----------------+--------------------------------------------------------+----
-----------------------------------+----------------------------------------+----------
+
|0.09              |0.62           |6             |294                 |4
  |0            |1   |0                   |accounting |low   |8.0     |0.0      |(9,
[8],[1.0])  |(2,[0],[1.0])   |(18,[0,1,2,3,4,15,16],[0.09,0.62,6.0,294.0,4.0,1.0,1.0])|
[-0.906113009228541,0.906113009228541]  |[0.28779589386338533,0.7122041061366147]|1.0
```

```
    |
|0.09              |0.62            |6              |294                 |4
  |0             |1   |0                    |accounting |low   |8.0     |0.0       |(9,
[8],[1.0])  |(2,[0],[1.0])   |(18,[0,1,2,3,4,15,16],[0.09,0.62,6.0,294.0,4.0,1.0,1.0])|
[-0.906113009228541,0.906113009228541]  |[0.28779589386338533,0.7122041061366147]|1.0
    |
|0.09              |0.77            |5              |275                 |4
  |0             |1   |0                    |product_mng|medium|4.0     |1.0       |(9,
[4],[1.0])  |(2,[1],[1.0])   |(18,[0,1,2,3,4,11,17],[0.09,0.77,5.0,275.0,4.0,1.0,1.0])|
[-0.6706336468375675,0.6706336468375675]|[0.3383549711925885,0.6616450288074115] |1.0
    |
|0.09              |0.77            |6              |244                 |4
  |0             |1   |0                    |product_mng|low   |4.0     |0.0       |(9,
[4],[1.0])  |(2,[0],[1.0])   |(18,[0,1,2,3,4,11,16],[0.09,0.77,6.0,244.0,4.0,1.0,1.0])|
[-0.7998272517122431,0.7998272517122431]|[0.31006247261884357,0.6899375273811564]|1.0
    |
+------------------+---------------+--------------+--------------------+----------------
--+-------------+----+--------------------+-----------+------+--------+---------+------
---------+----------------+---------------------------------------------------------+----
------------------------------------+----------------------------------------+----------
+
only showing top 4 rows
```

Evaluation of Decision Tree Classifier Model

In [ ]:

```python
evaluation = ["f1","accuracy","weightedPrecision","weightedRecall", "weightedTruePositiv
for i in evaluation:
  from pyspark.ml.evaluation import MulticlassClassificationEvaluator
  eval = MulticlassClassificationEvaluator(predictionCol="prediction", labelCol= "left",
  print(i, ":", eval.evaluate(result2))
```

```
f1 : 0.7779568868738669
accuracy : 0.801658604008293
weightedPrecision : 0.7820065880658045
weightedRecall : 0.8016586040082929
weightedTruePositiveRate : 0.8016586040082929
weightedFalsePositiveRate : 0.5166086426447655
weightedFMeasure : 0.7779568868738669
truePositiveRateByLabel : 0.9422208847427024
falsePositiveRateByLabel : 0.6571709233791748
precisionByLabel : 0.8239473684210527
recallByLabel : 0.9422208847427024
fMeasureByLabel : 0.8791239646216482
logLoss : 0.4152158353029982
hammingLoss : 0.19834139599170697
```

In [ ]: